Aplikacja MusicApp

Projekt na przedmiot Zaawansowane techniki programistyczne

Wykonawcy:

Viktora Kopasovska
Adrian Łupiński
Pavel Nasytka

Grupa PS8

1 Wstęp

Założeniem aplikacji było stworzenie własnego odtwarzacza muzycznego. Celem projektantów było stworzenie maksymalnie prostej w obsłudze i czytelnej aplikacji.

2. Analiza wymagań systemu:

2.1 Wymagania funkcjonalne:

Użytkownik aplikacji z jej pomocą może odtwarzać swoją kolekcje utworów. Utwór można w dowolnym momencie zatrzymać i wznowić od tego momentu. Do tego można dowolnie przechodzić po liście utworów. Użytkownik nie musi przejmować się awarią aplikacji ponieważ wznawia ona prace od ostatniego momentu w którym działała prawidłowo .

2.2 Wymagania niefunkcjonalne:

Użytkownik musi wyłącznie rozpakować swoją aplikację w dowolnie wybranym folderze . poruszanie ułatwia maksymalnie prosty i czytelny interfejs użytkownika. Do obsługi nie jest wymagana wiedza z zakresu programowania.

3 Wykorzystane technologie:

- Język programowania: c#
- API: WPF
- Zewnętrzna biblioteka LibTag#

4 Specyficzne rozwiązanie:

Na potrzeby projektu, okazało się niezbędnym rozwiązanie problemu odczytywania tagów z plików mp3 takich jak na przykład dane artysty czy albumu. Na pomoc przyszła zewnętrzna biblioteka TagLib# która to po zaimplementowaniu rozwiązała nurtujący programistów problem. Biblioteka zajmuje się odczytywaniem i zapisywaniem, metadanych z plików mp3. W znalezieniu rozwiązania problemu pomocna okazała się strona StackOverflow.com. Przy okazji dowiedzieliśmy się, że wcześniej podobne problemy spotkały wielu innych programistów.

Link do tematu który podpowiedział w jaki sposób rozwiązać problem :

https://stackoverflow.com/questions/13221413/how-do-i-read-mp3-tags

Link który pomógł nam w zainstalowaniu powyższej biblioteki:

https://stackoverflow.com/questions/40826094/how-do-i-use-taglib-sharp

5 Podział prac:

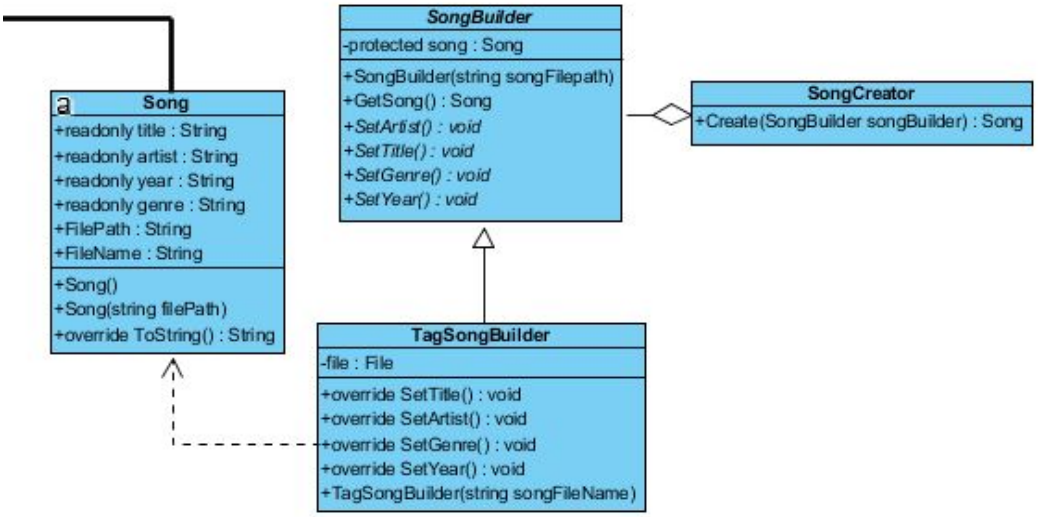
- Viktoria Kopasovska:
  Wykonanie ‘MainWindowViewModel’ oraz implementacja wzorca Builder

- Adrian Łupiński :
  Implementacja wzorca Memento oraz Command

- Pavel Nasytka :
  Implementacja wzorca Singleton oraz Iterator.

6. Zmiany w diagramie UML

- Builder

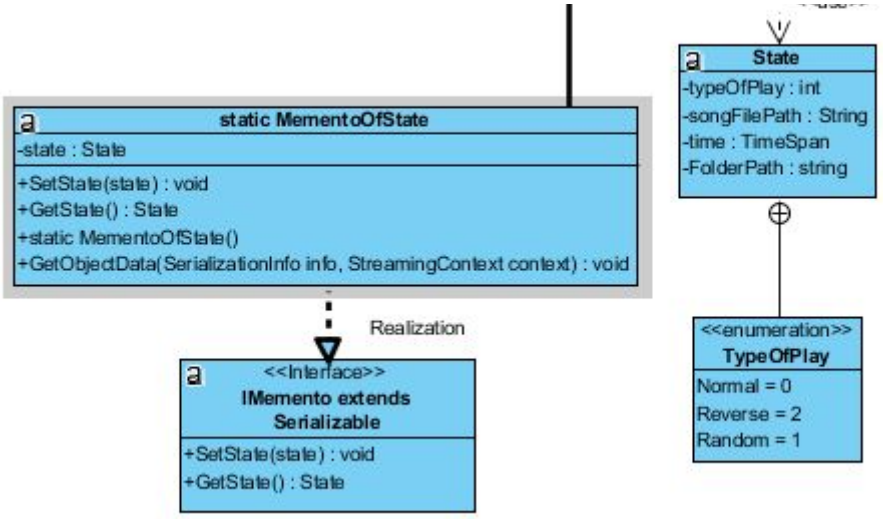


Przerobiliśmy wzorzec Builder. Wykorzystujemy go dla konstrukcji obiektu song z zachowaniem kolejności tworzenia składników (tytuł, autor, gatunek, rok). Podstawowy element to klasa abstrakcyjna SongBuilder, udostępnia metody budowy. Dziedziczy klasa TagSongBuilder. Właśnie ona decyduje , jak będzie tworzony obiekt. Klasa SongCreator jest stworzona po to, żeby kontrolować tworzenie utworu (uruchamia metody budowniczego w konkretnej kolejności).

- Iterator

W iteratorze dodaliśmy nową metodę Prev(). Pozwala przechodzić po liście utworów do tyłu.

- Memento



Dodaliśmy typ enum TypeOfPlay w klasie State. Zawiera trzy typy: Normal, Random i Reverse. Oraz została stworzona dodatkowa metoda GetObjectData. Pobiera informacje o obiekcie (czyli utworze).

7. Diagram UML w całości